Factoring $N = p^2q$

Nathan Manohar Ben Fisch

Abstract

1 Introduction and Problem Overview

The problem of factoring p^2q is of considerable interest in cryptography. Various encryption schemes, such as the EPOC cryptosystem [2], are based on the assumption that this problem is hard to solve. A natural question to ask is whether factoring p^2q is as hard as factoring a general RSA modulus $N = pq$. In particular, is there a way to exploit the fact that N is of the form p^2q that could lead to an improvement in factoring moduli of this form? One observation is that given $N = p^2q$, it is easy to learn if q is a quadratic residue modulo some prime ℓ. To see this, we first note that the Jacobi symbol $\left(\frac{\ell}{N}\right)$ is computable in polynomial time via repeated application of the law of quadratic reciprocity. Additionally, since the Jacobi symbol is a multiplicative function, it follows that

$$\left(\frac{\ell}{N}\right) = \left(\frac{\ell}{p^2q}\right) = \left(\frac{\ell}{p^2}\right)\left(\frac{\ell}{q}\right) = \left(\frac{\ell}{p}\right)^2\left(\frac{\ell}{q}\right).$$

Since $\gcd(\ell, p) = 1$ (for $\ell \neq p$), it follows that the Legendre symbol $\left(\frac{\ell}{p}\right) = \pm 1$ and therefore

$$\left(\frac{\ell}{N}\right) = \left(\frac{\ell}{q}\right).$$

By the law of quadratic reciprocity, it follows that

$$\left(\frac{\ell}{q}\right)\left(\frac{q}{\ell}\right) = (-1)^{\frac{\ell-1}{2}\frac{q-1}{2}},$$

and so $\left(\frac{q}{\ell}\right)$ can be computed in polynomial time.

Using this information, we can construct tables T_{ℓ_i} for primes ℓ_i that list the possible values of $q \bmod \ell_i$. Since there are $\frac{\ell_i-1}{2}$ quadratic residues and nonresidues modulo ℓ_i, the size of T_{ℓ_i} will be $\frac{\ell_i-1}{2}$. Equivalently, let $b_i = \left(\frac{q}{\ell_i}\right)$ then the table T_{ℓ_i} contains the roots of the polynomial $(x^{(\ell_i-1)/2} - b_i)$ over $\mathbb{Z}_{\ell_i}$. The question of factoring p^2q has now been reduced to the question of whether q can be efficiently reconstructed given information about whether or not q is a quadratic residue modulo primes for a fixed sequence of primes.

2 One Approach

One approach to factoring p^2q is to attempt to construct a polynomial that must have q as a small root and then determine this root using Coppersmith's method [1]. Observe that for each prime ℓ_i for which we have a corresponding table T_{ℓ_i}, we can construct the polynomial

$$f_{\ell_i}(x) = \prod_{a \in T_{\ell_i}} (x - a) \bmod \ell_i.$$

Then, using the Chinese remainder theorem, we can construct the polynomial $f \bmod \prod_i \ell_i$ obtained by applying to the Chinese remainder theorem term-wise to the coefficients of each of the f_{ℓ_i}'s. Note that since q is a root of all the f_{ℓ_i}'s, it follows that q is a root of f. However, q is not a sufficiently small root of f for Coppersmith's algorithm to determine it. In particular, if f is a polynomial of degree d modulo L, Coppersmith's algorithm will only find roots that are $< L^{1/d}$. However, if the ℓ_i's are the first n primes, then $\prod_i \ell_i \approx e^n$. Since we must have $\prod_i \ell_i > q$, it follows that we need $n > \ln q$. Additionally, we note that the degree of f will be the size of the largest table, which will be $\frac{\ell_n-1}{2}$. Since the size of the largest prime is $\approx n \ln n$, it follows that $L^{1/d}$ will be approximately

$$e^{2n/n\ln n} = e^{2/\ln n} = e^{O(1/\log\log q)} << q,$$

and so Coppersmith's method will not be able to determine q.

An immediate observation is that if the degree of f was smaller (was $n^{1-\varepsilon}$ instead of $\approx n \ln n$), then Coppersmith's method would be sufficient to determine q.

In this case, we would have that $L^{1/d}$ is $\approx e^{n^\varepsilon} > q$ for $n = O(\log^{1/\varepsilon} q)$. However, there does not seem to be any way to reduce the degree using these tables alone. In particular, since the size of the tables is $O(n \ln n)$, f must necessarily have degree $O(n \ln n)$ in order to capture all the possible values of q. If some extra information about q unrelated to the quadratic residuosity of q modulo primes could be gathered, this could potentially be leveraged to reduce the size of the tables.

3 Factoring $N = p^r q$

4 General Purpose Factoring Algorithms

5 Beyond Coppersmith's method

Given that Coppersmith's method will not work for finding sufficiently many roots (i.e., more than constant size roots) of a polynomial of degree O(log(N)) over N, we need to search for other methods. Define $K = p_1 \cdots p_k$ such that $q < K$. Define $N = p_1 \cdots p_n$ for $n > k$. As before, let F be the unique polynomial mod N that is equivalent to $f_i = (x^{(p_i-1)/2} - b_i)$ over $\mathbb{Z}_{p_i}$ where $b_i = \left(\frac{q}{p_i}\right)$. The first question is whether we can even hope that there will be a small (i.e. polynomial) number of roots of F that are less than K, otherwise there would probably be no hope of recovering q. When $n = k$ it is easy to see that there are an exponential number of roots in $[0, K)$ because any k-wise combination of the roots of f_i for each i corresponds to a unique integer in $\mathbb{Z}_k$ that is a root of F. This total number of roots is precisely $\Pi_{i=1}^{k}(p_i - 1)/2 \approx K/2^k$. Although the number of solutions for q remaining is still exponential, the information about q that we got from examining the roots of $f_1, ..., f_k$ reduced the solution space by a factor 2^k. Heuristically, we would hope that each polynomial f_i for $i > k$ would continue to reduce the solution space by a factor 2 so that we would only need $n = O(k) = O(log(q))$ polynomials to reduce the number of solutions to a constant. However, proving this seems to be tricky and we are working on a proof.

6 Turning the Problem Around

Another idea we are working on is to turn this problem around and assume it is hard to factor $p^2 q$, and thus that it is hard to find small roots of polynomials of poly degree over a super smooth modulus N and to build some useful crypto primitives like key exchange from this assumption. This problem may also be quantum hard.

7 Conclusions/Open Problems

Since this problem generally reduces to factoring polynomials modulo composites, if time permits, we may survey some of the known factoring algorithms in this space. One such problem is that of factoring polynomials over finite fields, for which polynomial time algorithms exist. Von Zur Gathen and Panario wrote a nice survey [3] summarizing the main methods for factoring univariate polynomials over finite fields.

References

[1] COPPERSMITH, D. Finding a small root of a univariate modular equation. In *Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques* (Berlin, Heidelberg, 1996), EUROCRYPT'96, Springer-Verlag, pp. 155–165.

[2] OKAMOTO, T., UCHIYAMA, S., AND FUJISAKI, E. Epoc: Efficient probabilistic public-key encryption (submission to p1363a).

[3] VON ZUR GATHEN, J., AND PANARIO, D. Factoring polynomials over finite fields: A survey. *Journal of Symbolic Computation 31*, 1 (2001), 3 – 17.